(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 August 2002 (08.08.2002)

PCT

(10) International Publication Number
WO 02/061525 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number: PCT/US01/45772

(22) International Filing Date:
2 November 2001 (02.11.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/245,295 2 November 2000 (02.11.2000) US
60/301,378 27 June 2001 (27.06.2001) US

(71) Applicant *(for all designated States except US)*: **PIRUS NETWORKS** [US/US]; 43 Nagog Park, Acton, MA 01720-3425 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **HALL, Howard** [/US]; 86 Mara Lande, Groton, MA 01450 (US). **SOKOLINSKI, Ilia** [/US]; 49 Union Street, Brighton, MA 02135 (US). **MERHAR, Milan** [/US]; 697 Boylston Street, Brookline, MA 02445 (US).

(74) Agent: **JACOBS, David**; Lucash, Gesmer & Updegrove, LLP, 40 Broad Street, Boston, MA 02109 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 02/061525 A2

(54) Title: TCP/UDP ACCELERATION

(57) Abstract: Disclosed are improved methods, devices and systems for storage management in digital networks.

SWITCHING SYSTEM

Incorporation by Reference/Priority Claim

Commonly owned U.S. provisional application for patent Serial No. 60/245,295 filed November 2, 2000, incorporated by reference herein; and

Commonly owned U.S. provisional application for patent Serial No. 60/301,378 filed June 27, 2001, incorporated by reference herein.

Additional publications are incorporated by reference herein as set forth below.

Field of the Invention

The present invention relates to digital information processing, and particularly to methods, systems and protocols for managing storage in digital networks.

Background of the Invention

The rapid growth of the Internet and other networked systems has accelerated the need for processing, transferring and managing data in and across networks.

In order to meet these demands, enterprise storage architectures have been developed, which typically provide access to a physical storage pool through multiple independent SCSI channels interconnected with storage via multiple front-end and back-end processors/controllers. Moreover, in data networks based on IP/Ethernet technology, standards have been developed to facilitate network management. These standards include Ethernet, Internet Protocol (IP), Internet Control Message Protocol (ICMP), Management Information Block (MIB) and Simple Network Management Protocol (SNMP). Network Management Systems (NMSs) such as HP Open View utilize these standards to

discover and monitor network devices. Examples of networked architectures are disclosed in the following patent documents, the disclosures of which are incorporated herein by reference:

US 5,941,972	Crossroads Systems, Inc.
US 6,000,020	Gadzoox Network, Inc.
US 6,041,381	Crossroads Systems, Inc.
US 6,061,358	McData Corporation
US 6,067,545	Hewlett-Packard Company
US 6,118,776	Vixel Corporation
US 6,128,656	Cisco Technology, Inc.
US 6,138,161	Crossroads Systems, Inc.
US 6,148,421	Crossroads Systems, Inc.
US 6,151,331	Crossroads Systems, Inc.
US 6,199,112	Crossroads Systems, Inc.
US 6,205,141	Crossroads Systems, Inc.
US 6,247,060	Alacritech, Inc.
WO 01/59966	Nishan Systems, Inc.

Conventional systems, however, do not enable seamless connection and interoperability among disparate storage platforms and protocols. Storage Area Networks (SANs) typically use a completely different set of technology based on Fibre Channel (FC) to build and manage storage networks. This has led to a "re-inventing of the wheel" in many cases. Users are often require to deal with multiple suppliers of routers, switches, host bus adapters and other components, some of which are not well-adapted to communicate with one another. Vendors and standards bodies continue to determine the protocols to be used to interface devices in SANs and NAS configurations; and SAN devices do not integrate well with existing IP-based management systems.

Still further, the storage devices (Disks, RAID Arrays, and the like), which are Fibre Channel attached to the SAN devices, typically do not support IP (and the SAN devices have limited IP support) and the storage devices cannot be discovered/managed by IP-based management systems. There are essentially two sets of management products – one for the IP devices and one for the storage devices.

Accordingly, it is desirable to enable servers, storage and network-attached storage (NAS) devices, IP and Fibre Channel switches on

storage-area networks (SAN), WANs or LANs to interoperate to provide improved storage data transmission across enterprise networks.

In addition, among the most widely used protocols for communications within and among networks, TCP/IP (TCP/Internet Protocol) is the suite of communications protocols used to connect hosts on the Internet. TCP provides reliable, virtual circuit, end-to-end connections for transporting data packets between nodes in a network. Implementation examples are set forth in the following patent and other publications, the disclosures of which are incorporated herein by reference:

US 5,260,942	IBM
US 5,442,637	ATT
US 5,566,170	Storage Technology Corporation
US 5,598,410	Storage Technology Corporation
US 5,598,410	Storage Technology Corporation
US 6,006,259	Network Alchemy, Inc.
US 6,018,530	Sham Chakravorty
US 6,122,670	TSI Telsys, Inc.
US 6,163,812	IBM
US 6,178,448	IBM

"TCP/IP Illustrated Volume 2", Wright, Stevens;
"SCSI over TCP", IETF draft, IBM, CISCO, Sangate, February 2000;
"The SCSI Encapsulation Protocol (SEP)", IETF draft, Adaptec Inc., May 2000;
RFC 793 "Transmission Control Protocol", September 1981.

Although TCP is useful, it requires substantial processing by the system CPU, thus limiting throughput and system performance. Designers have attempted to avoid this limitation through various inter-processor communications techniques, some of which are described in the above-cited publications. For example, some have offloaded TCP processing tasks to an auxiliary CPU, which can reside on an intelligent network interface or similar device, thereby reducing load on the system CPU. However, this approach does not eliminate the problem, but merely moves it elsewhere in the system, where it remains a single chokepoint of performance limitation.

Others have identified separable components of TCP processing and implemented them in specialized hardware. These can include calculation or verification of TCP checksums over the data being transmitted, and the appending or removing of fixed protocol headers to or from such data. These approaches are relatively simple to implement in hardware to the extent they perform only simple, condition-invariant manipulations, and do not themselves cause a change to be applied to any persistent TCP state variables. However, while these approaches somewhat reduce system CPU load, they have not been observed to provide substantial performance gains.

Some required components of TCP, such as retransmission of a TCP segment following a timeout, are difficult to implement in hardware, because of their complex and condition-dependent behavior. For this reason, systems designed to perform substantial TCP processing in hardware often include a dedicated CPU capable of handling these exception conditions. Alternatively, such systems may decline to handle TCP segment retransmission or other complex events and instead defer their processing to the system CPU.

However, a major difficulty in implementing such "fast path/slow path" solutions is ensuring that the internal state of the TCP connections, which can be modified as a result of performing these operations, is consistently maintained, whether the operations are performed by the "fast path" hardware or by the "slow path" system CPU.

It is therefore desirable to provide methods, devices and systems that simplify and improve these operations.

It is also desirable to provide methods, devices and systems that simplify management of storage in digital networks, and enable flexible deployment of NAS, SAN and other storage systems, and Fibre Channel (FC), IP/Ethernet and other protocols, with storage subsystem and location independence.

Summary of the Invention

The invention addresses the noted problems typical of prior art systems, and in one aspect, provides a switch system having a first configurable set of processor elements to process storage resource connection requests, a second configurable set of processor elements capable of communications with the first configurable set of processor elements to receive, from the first configurable set of processor elements, storage connection requests representative of client requests, and to route the requests to at least one of the storage elements, and a configurable switching fabric interconnected between the first and second sets of processor elements, for receiving at least a first storage connection request from one of the first set of processor elements, determining an appropriate one of the second set of processors for processing the storage connection request, automatically configuring the storage connection request in accordance with a protocol utilized by the selected one of the second set of processors, and forwarding the storage connection request to the selected one of the second set of processors for routing to at least one of the storage elements.

Another aspect of the invention provides methods, systems and devices for enabling data replication under NFS servers.

A further aspect of the invention provides mirroring of NFS servers using a multicast function.

Yet another aspect of the invention provides dynamic content replication under NFS servers.

In another aspect, the invention provides load balanced NAS using a hashing or similar function, and dynamic data grooming and NFS load balancing across NFS servers.

The invention also provides, in a further aspect, domain sharing across multiple FC switches, and secure virtual storage domains (SVSD).

Still another aspect of the invention provides TCP/UDP acceleration, with IP stack bypass using a network processors (NP). The present invention simultaneously maintaining TCP state information in both the fast path and the slow path. Control messages are exchanged

between the fast path and slow path processing engines to maintain state synchronization, and to hand off control from one processing engine to another. These control messages can be optimized to require minimal processing in the slow path engines (e.g., system CPU) while enabling efficient implementation in the fast path hardware. This distributed synchronization approach significantly accelerates TCP processing, but also provides additional benefits, in that it permits the creation of more robust systems.

The invention, in another aspect, also enables automatic discovery of SCSI devices over an IP network, and mapping of SNMP requests to SCSI.

In addition, the invention also provides WAN mediation caching on local devices.

Each of these aspects will next be described in detail, with reference to the attached drawing figures.

Brief Description of the Drawings

FIG. 1 depicts a hardware architecture of one embodiment of the switch system aspect of the invention.

FIG. 2 depicts interconnect architecture useful in the embodiment of FIG. 1.

FIG. 3 depicts processing and switching modules.

FIG. 4 depicts software architecture in accordance with one embodiment of the invention.

FIG. 5 depicts detail of the client abstraction layer.

FIG. 6 depicts the storage abstraction layer.

FIG. 7 depicts scaleable NAS.

FIG. 8 depicts replicated local/remote storage.

FIG. 9 depicts a software structure useful in one embodiment of the invention.

FIG. 10 depicts system services.

FIG. 11 depicts a management software overview.

FIG. 12 depicts a virtual storage domain.

FIG. 13 depicts another virtual storage domain.

FIG. 14 depicts configuration processing boot-up sequence.

FIG. 15 depicts a further virtual storage domain example.

FIG. 16 is a flow chart of NFS mirroring and related functions.

FIG. 17 depicts interface module software.

FIG. 18 depicts an flow control example.

FIG. 19 depicts hardware in an SRC.

FIG. 20 depicts SRC NAS software modules.

FIG. 21 depicts SCSI/UDP operation.

FIG. 22 depicts SRC software storage components.

FIG. 23 depicts FC originator/FC target operation.

FIG. 24 depicts load balancing NFS client requests between NFS servers.

FIG. 25 depicts NFS receive micro-code flow.

FIG. 26 depicts NFS transmit micro-code flow.

FIG. 27 depicts file handle entry into multiple server lists.

FIG. 28 depicts a sample network configuration in another embodiment of the invention.

FIG. 29 depicts an example of a virtual domain configuration.

FIG. 30 depicts an example of a VLAN configuration.

FIG. 31 depicts a mega-proxy example.

FIG. 32 depicts device discovery in accordance with another aspect of the invention.

FIG. 33 depicts SNMP/SCSI mapping.

FIG. 34 SCSI response/SNMP trap mapping.

FIG. 35 depicts data structures useful in another aspect of the invention.

FIG. 36 depicts mirroring and load balancing operation.

FIG. 37 depicts server classes.

FIGS. 38A, 38B, 38C depict mediation configurations in accordance with another aspect of the invention.

FIG. 39 depicts operation of mediation protocol engines.

FIG. 40 depicts configuration of storage by the volume manager in accordance with another aspect of the invention.

FIG. 41 depicts data structures for keeping track of virtual devices and sessions.

FIG. 42 depicts mediation manager operation in accordance with another aspect of the invention.

FIG. 43 depicts mediation in accordance with one practice of the invention.

FIG. 44 depicts mediation in accordance with another practice of the invention.

FIG. 45 depicts fast-path architecture in accordance with the invention.

FIG. 46 depicts IXP packet receive processing for mediation.

Detailed Description of the Invention

I. Overview

FIG. 1 depicts the hardware architecture of one embodiment of a switch system according to the invention. As shown therein, the switch system 100 is operable to interconnect clients and storage. As discussed in detail below, storage processor elements 104 (SPs) connect to storage; IP processor elements 102(IP) connect to clients or other devices; and a high speed switch fabric 106 interconnects the IP and SP elements, under the control of control elements 103.

The IP processors provide content-aware switching, load balancing, mediation, TCP/UDP hardware acceleration, and fast forwarding, all as discussed in greater detail below. In one embodiment, the high speed fabric comprises redundant control processors and a redundant switching fabric, provides scalable port density and is media-independent. As described below, the switch fabric enables media-independent module interconnection, and supports low-latency Fibre Channel (F/C) switching. In an embodiment of the invention commercially

available from the assignee of this application, the fabric maintains QoS for Ethernet traffic, is scalable from 16 to 256 Gbps, and can be provisioned as fully redundant switching fabric with fully redundant control processors, ready for 10 Gb Ethernet, InfiniBand and the like. The SPs support NAS (NFS/CIFS), mediation, volume management, Fibre Channel (F/C) switching, SCSI and RAID services.

FIG. 2 depicts an interconnect architecture adapted for use in the switching system 100 of FIG. 1. As shown therein, the architecture includes multiple processors interconnected by dual paths 110, 120. Path 110 is a management and control path adapted for operation in accordance with switched Ethernet. Path 120 is a high speed switching fabric, supporting a point to point serial interconnect. Also as shown in FIG. 2, front-end processors include SFCs 130, LAN Resource Cards (LRCs) 132, and Storage Resource Cards (SRCs) 134, which collectively provide processing power for the functions described below. Rear-end processors include MICs 136, LIOs 138 and SIOs 140, which collectively provide wiring and control for the functions described below.

In particular, the LRCs provide interfaces to external LANs, servers, WANs and the like (such as by 4 x Gigabit Ethernet or 32 x 10/100 Base-T Ethernet interface adapters); perform load balancing, content-aware switching of internal services; implement storage mediation protocols; and provide TCP hardware acceleration.

The SRCs interface to external storage or other devices (such as via Fibre Channel, 1 or 2 Gbps, FC-AL or FC-N)

As shown in FIG. 3, LRCs and LIOs are network processors providing LAN-related functions. They can include GBICs and RJ45 processors. MICs provide control and management. As discussed below, the switching system utilizes redundant MICs and redundant fabrics. The FIOs shown in FIG. 3 provide F/C switching. These modules can be commercially available ASIC-based F/C switch elements, and collectively enable low cost, high-speed SAN using the methods described below.

FIG. 4 depicts a software architecture adapted for use in an embodiment of switching system 100, wherein a management layer 402 interconnects with client services 404, mediation services 406, storage services 408, a client abstraction layer 410, and a storage abstraction layer 412. In turn, the client abstraction layer interconnects with client interfaces (LAN, SAN or other) 414, and the storage abstraction layer interconnects with storage devices or storage interfaces (LAN, SAN or other) 416.

The client abstraction layer isolates, secures, and protects internal resources; enforces external group isolation and user authentication; provides firewall access security; supports redundant network access with fault failover, and integrates IP routing and multiport LAN switching. It addition, it presents external clients with a "virtual service" abstraction of internal services, so that there is no need to reconfigure clients when services are changed. Further, it provides internal services a consistent network interface, wherein service configuration is independent of network connectivity, and there is no impact from VLAN topology, multihoming or peering.

FIG. 5 provides detail of the client abstraction layer. As shown therein, it can include TCP acceleration function 502 (which, among other activities, offloads processing reliable data streams); load balancing function 504 (which distributes requests among equivalent resources); content-aware switching 506 (which directs requests to an appropriate resource based on the contents of the requests/packets); virtualization function 508 (which provides isolation and increased security); 802.1 switching and IP routing function 510 (which supports link/path redundancy), and physical I/F support functions 512 (which can support 10/100Base-T, Gigabit Ethernet, Fibre Channel and the like).

In addition, an internal services layer provides protocol mediation, supports NAS and switching and routing. In particular, in iSCSI applications the internal services layer uses TCP/IP or the like to provide LAN-attached servers with access to block-oriented storage; in FC/IP it interconnects Fibre Channel SAN "islands" across an Internet backbone;

and in IP/FC applications it extends IP connectivity across Fibre Channel. Among NAS functions, the internal services layer includes support for NFS (industry-standard Network File Service, provided over UDP/IP (LAN) or TCP/IP (WAN); and CIFS (compatible with Microsoft Windows File Services, also known as SMB. Among switching and routing functions, the internal services layer supports Ethernet, Fibre Channel and the like.

The storage abstraction layer shown in FIG. 6 includes file system 602, volume management 604, RAID function 606, storage access processing 608, transport processing 610 an physical I/F support 612. File system layer 602 supports multiple file systems; the volume management layer creates and manages logical storage partitions; the RAID layer enables optional data replication; the storage access processing layer supports SCSI or similar protocols, and the transport layer is adapted for Fibre Channel or SCSI support. The storage abstraction layer consolidates external disk drives, storage arrays and the like into a sharable, pooled resource; and provides volume management that allows dynamically resizeable storage partitions to be created within the pool; RAID service that enables volume replication for data redundancy, improved performance; and file service that allows creation of distributed, sharable file systems on any storage partition.

A technical advantage of this configuration is that a single storage system can be used for both file and block storage access (NAS and SAN).

FIGS. 7 and 8 depict examples of data flows through the switching system 100. (It will be noted that these configurations are provided solely by way of example, and that other configurations are possible.) In particular, as will be discussed in greater detail below, FIG. 7 depicts a scaleable NAS example, while FIG. 8 depicts a replicated local/remote storage example. As shown in FIG. 7, the switch system 100 includes secure virtual storage domain (SVSD) management layer 702, NFS servers collectively referred to by numeral 704, and modules 706 and 708.

Gigabit module 706 contains TCP 710, load balancing 712, content-aware switching 714, virtualization 716, 802.1 switching and IP routing 718, and Gigabit (GV) optics collectively referred to by numeral 720.

FC module 708 contains file system 722, volume management 724, RAID 726, SCSI 728, Fibre Channel 730, and FC optics collectively referred to by numeral 731.

As shown in the scaleable NAS example of FIG. 7, the switch system 100 connects clients on multiple Gigabit Ethernet LANs 732 (or similar) to (1) unique content on separate storage 734 and replicated filesystems for commonly accessed files 736. The data pathways depicted run from the clients, through the GB optics, 802.1 switching and IP routing, virtualization, content-aware switching, load balancing and TCP, into the NFS servers (under the control/configuration of SVSD management), and into the file system, volume management, RAID, SCSCI, Fibre Channel, and FC optics to the unique content (which bypasses RAID), and replicated filesystems (which flows through RAID).

Similar structures are shown in the replicated local/remote storage example of FIG. 8. However, in this case, the interconnection is between clients on Gigagbit Ethernet LAN (or similar) 832, secondary storage at an offsite location via a TCP/IP network 834, and locally attached primary storage 836. In this instance, the flow is from the clients, through the GB optics, 802.1 switching and IP routing, virtualization, content-aware switching, load balancing and TCP, then through iSCSI mediation services 804 (under the control/configuration of SVSD management 802), then through volume management 824, and RAID 826. Then, one flow is from RAID 826 through SCSI 828, Fibre Channel 830 and FC Optics 831 to the locally attached storage 836; while another flow is from RAID 826 back to TCP 810, load balancing 812, content-aware switching 814, virtualization 816, 802.1 switching and IP routing 818 and GB optics 820 to secondary storage at an offsite location via a TCP/IP network 834.

II. Hardware/Software Architecture

This section provides an overview of the structure and function of the invention (alternatively referred to hereinafter as the "Pirus box"). In one embodiment, the Pirus box is a 6 slot, carrier class, high performance, multi-layer switch, architected to be the core of the data storage infrastructure. The Pirus box will be useful for ASPs (Application Storage Providers), SSPs (Storage Service Providers) and large enterprise networks. One embodiment of the Pirus box will support Network Attached Storage (NAS) in the form of NFS attached disks off of Fibre Channel ports. These attached disks are accessible via 10/100/1000 switched Ethernet ports. The Pirus box will also support standard layer 2 and Layer 3 switching with port-based VLAN support, and layer 3 routing (on unlearned addresses). RIP will be one routing protocol supported, with OSPF and others also to be supported. The Pirus box will also initiate and terminate a wide range of SCSI mediation protocols, allowing access to the storage media either via Ethernet or SCSI/FC. The box is manageable via a CLI, SNMP or an HTTP interface.

1 Software Architecture Overview

FIGURE 9 is a block diagram illustrating the software modules used in the Pirus box (the terms of which are defined in the glossary set forth below). As shown in FIG. 9, the software structures correspond to MIC 902, LIC 904, SRC-NAS 908 and SRC-Mediator 910, interconnected by MLAN 905 and fabric 906. The operation of each of the components shown in the drawing is discussed below.

1.1 System Services

The term System Service is used herein to denote a significant function that is provided on every processor in every slot. It is contemplated that many such services will be provided; and that they can be segmented into 2 categories: 1) abstracted hardware services and 2) client/server services. The attached FIGURE 10 is a diagram of some of the exemplary interfaces. As shown in FIG. 10, the system services correspond to IPCs 1002 and 1004 associated with fabric and control channel 1006, and with services SCSI 1008, RSS 1010, NPCS 1012, AM 1014, Log/Event 1016, Cache/Bypass 1018, TCP/IP 1020, and SM 1022.

1.1.1 SanStreaM (SSM) System Services (S2)

SSM system service can be defined as a service that provides a software API layer to application software while "hiding" the underlying hardware control. These services may add value to the process by adding protocol layering or robustness to the standard hardware functionality.

System services that are provided include:

Card Processor Control Manager (CPCM). This service provides a mechanism to detect and manage the issues involved in controlling a Network Engine Card (NEC) and its associated Network Processors (NP). They include insertion and removal, temperature control, crash management, loader, watchdog, failures etc.

Local Hardware Control (LHC). This controls the hardware local to the board itself. It includes LEDS, fans, and power.

Inter-Processor Communication (IPC). This includes control bus and fabric services, and remote UART.

1.1.2 SSM Application Service (AS)

Application services provide an API on top of SSM system services. They are useful for executing functionality remotely. Application Services include:

Remote Shell Service (RSS) – includes redirection of debug and other valuable info to any pipe in the system.

Statistics Provider – providers register with the stats consumer to provide the needed information such as mib read only attributes.

Network Processor Config Service (NPCS) – used to receive and process configuration requests.

Action Manager – used to send and receive requests to execute remote functionality such as rebooting, clearing stats and re-syncing with a file system.

Logging Service – used to send and receive event logging information.

Buffer Management – used as a fast and useful mechanism for allocating, typing, chaining and freeing message buffers in the system.

HTTP Caching/Bypass service – sub-system to supply an API and functional service for HTTP file caching and bypass. It will make the determination to cache a file, retrieve a cached file (on board or off), and bypass a file (on board or not). In addition this service will keep track of local cached files and their associated TTL, as well as statistics on file bypassing. It will also keep a database of known files and their caching and bypassing status.

Multicast services – A service to register, send and receive multicast packets across the MLAN.

2. Management Interface Card

The Management Interface Card (MIC) of the Pirus box has a single high performance microprocessor and multiple 10/100 Ethernet interfaces for administration of the SANStream management subsystem. This card also has a PCMCIA device for bootstrap image and configuration storage.

In the illustrated embodiments, the Management Interface Card will not participate in any routing protocol or forwarding path decisions. The IP stack and services of VxWorks will be used as the underlying IP facilities for all processes on the MIC. The MIC card will also have a flash based, DOS file system.

The MIC will not be connected to the backplane fabric but will be connected to the MLAN (Management LAN) in order to send/receive data to/from the other cards in the system. The MLAN is used for all MIC □□ "other cards" communications.

2.1. Management Software

Management software is a collection of components responsible for configuration, reporting (status, statistics, etc), notification (events) and billing data (accounting information). The management software may also include components that implement services needed by the other modules in the system.

Some of the management software components can exist on any processor in the system, such as the logging server. Other components reside only on the MIC, such as the WEB Server providing the WEB user interface.

The strategy and subsequent architecture must be flexible enough to provide a long-term solution for the product family. In other words, the 1.0 implementation must not preclude the inclusion of additional management features in subsequent releases of the product.

The management software components that can run on either the MIC or NEC need to meet the requirement of being able to "run anywhere" in the system.

2.2 Management Software Overview

In the illustrated embodiments the management software decomposes into the following high-level functions, shown in FIGURE 11. As shown in the example of FIG. 11 (other configurations are also possible and within the scope of the invention), management software can be organized into User Interfaces (UIs) 1102, rapid control backplane (RCB) data dictionary 1104, system abstraction model (SAM) 1106, configuration & statistics manager (CSM) 1108, and logging/billing APIs 1110, on module 1101. This module can communicate across system services (S2) 1112 and hardware elements 1114 with configuration & statistics agent (CSA) 1116 and applications 1118.

The major components of the management software include the following:

2.2.1 User Interfaces (UIs)

These components are the user interfaces that allow the user access to the system via a CLI, HTTP Client or SNMP Agent.

2.2.2 Rapid Control Backplane (RCB)

These components make up the database or data dictionary of settable/gettable objects in the system. The UIs use "Rapid Marks"

(keys) to reference the data contained within the database. The actual location of the data specified by a Rapid Mark may be on or off the MIC.

2.2.3 System Abstraction Model (SAM)

These components provide a software abstraction of the physical components in the system. The SAM works in conjunction with the RCB to get/set data for the UIs. The SAM determines where the data resides and if necessary interacts with the CSM to get/set the data.

2.2.4 Configuration & Statistics Manager (CSM)

These components are responsible for communicating with the other cards in the system to get/set data. For example the CSM sends configuration data to a card/processor when a UI initiates a change and receives statistics from a card/processor when a UI requests some data.

2.2.5 Logging / Billing APIs

These components interface with the logging and event servers provided by System Services and are responsible for sending logging/billing data to the desired location and generating SNMP traps/alerts when needed.

2.2.6 Configuration & Statistics Agent (CSA)

These components interface with the CSM on the MIC and responds to CSM messages for configuration/statistics data.

2.3 Dynamic Configuration

The SANStream management system will support dynamic configuration updates. A significant advantage is that it will be unnecessary to reboot the entire chassis when an NP's configuration is modified. The bootstrap configuration can follow similar dynamic guidelines. Bootstrap configuration is merely dynamic configuration of an NP that is in the reset state.

Both soft and hard configuration will be supported. Soft configuration allows dynamic modification of current system settings.

Hard configuration modifies bootstrap or start-up parameters. A hard configuration is accomplished by saving a soft configuration. A hard configuration change can also be made by (T)FTP of a configuration file. The MIC will not support local editing of configuration files.

In a preferred practice of the invention DNS services will be available and utilized by MIC management processes to resolve hostnames into IP addresses.

2.4 Management Applications

In addition to providing "rote" management of the system, the management software will be providing additional management applications/functions. The level of integration with the WEB UI for these applications can be left to the implementer. For example the Zoning Manager could be either be folded into the HTML pages served by the embedded HTTP server OR the HTTP server could serve up a stand-alone JAVA Applet.

2.4.1 Volume Manager

A preferred practice of the invention will provide a volume manager function. Such a Volume Manager may support:

- Raid 0 – Striping
- Raid 1 – Mirroring
- Hot Spares
- Aggregating several disks into a large volume.
- Partitioning a large disk into several smaller volumes.

1.4.2 Load Balancer

This application configures the load balancing functionality. This involves configuring policies to guide traffic through the system to its ultimate destination. This application will also report status and usage statistics for the configured policies.

1.4.3 Server-less Backup (NDMP)

This application will support NDMP and allow for serverless back up. This will allow users the ability to back up disk devices to tape devices without a server intervening.

2.4.4 IP-ized Storage Management

This application will "hide" storage and FC parameters from IP-centric administrators. For example, storage devices attached to FC ports will appear as IP devices in an HP-OpenView network map. These devices will be "ping-able", "discoverable" and support a limited scope of MIB variables.

In order to accomplish this IP addresses be assigned to the storage devices (either manually or automatically) and the MIC will have to be sent all IP Mgmt (exact list TBD) packets destined for one of the storage IP addresses. The MIC will then mediate by converting the IP packet (request) to a similar FC/SCSI request and sending it to the device.

For example an IP Ping would become a SCSI Inquiry while a SNMP get of sysDescription would also be a SCSI Inquiry with some of the returned data (from the Inquiry) mapped into the MIB variable and

returned to the requestor. These features are discussed in greater detail in the IP Storage Management section below.

2.4.5 Mediation Manager

This application is responsible for configuring, monitoring and managing the mediation between storage and networking protocols. This includes session configurations, terminations, usage reports, etc. These features are discussed in greater detail in the Mediation Manager section below.

2.4.6 VLAN Manager

Port level VLANs will be supported. Ports can belong to more than one VLAN.

The VLAN Manager and Zoning Manager could be combined into a VDM (or some other name) Manager as a way of unifying the Ethernet and FC worlds.

2.4.7 File System Manager

The majority of file system management will probably be to "accept the defaults". There may be an exception if it is necessary to format disks when they are attached to a Pirus system or perform other disk operations.

2.5 Virtual Storage Domain (VSD)

Virtual storage domains serve 2 purposes.

1. Logically group together a collection of resources.
2. Logically group together and "hide" a collection of resources from the outside world.

The 2 cases are very similar. The second case is used when we are load balancing among NAS servers.

FIGURE 12 illustrates the first example:

In this example Server 1 is using SCSI/IP to communicate to Disks A and B at a remote site while Server 2 is using SCSI/IP to communicate with Disks C and D at the same remote site. For this configuration Disks A, B, C, and D must have valid IP addresses. Logically inside the PIRUS system 2 Virtual Domains are created, one for Disks A and B and one for

Disks C and D. The IFF software doesn't need to know about the VSDs since the IP addresses for the disks are valid (exportable) it can simply forward the traffic to the correct destination. The VSD is configured for the management of the resources (disks).

The second usage of virtual domains is more interesting. In this case let's assume we want to load balance among 3 NAS servers. A VSD would be created and a Virtual IP Address (VIP) assigned to it. External entities would use this VIP to address the NAS and internally the PIRUS system would use NAT and policies to route the request to the correct NAS server. FIGURE 13 illustrates this.

In this example users of the NAS service would simple reference the VIP for Joe's ASP NAS LB service. Internally, through the combination of virtual storage domains and policies the Pirus system load balances the request among 3 internal NAS servers, thus providing a scalable, redundant NAS solution.

Virtual Domains can be use to virtualize the entire Pirus system. Within VSDs the following entities are noteworthy:

2.5.1 Services

Services represent the physical resources. Examples of services are:

1. Storage Devices attached to FC or Ethernet ports. These devices can be simple disks, complex RAID arrays, FC-AL connections, tape devices, etc.
2. Router connections to the Internet.
3. NAS - Internally defined ones only.

2.5.2 Policies

A preferred practice of the invention can implement the following types of policies:

1. Configuration Policy – A policy to configure another policy or a feature. For example a NAS Server in a virtual domain will be configured as a "Service". Another way to look at it is that a

Configuration Policy is simply the collection of configurable parameters for an object.

2. Usage Policy – A policy to define how data is handled. In our case load balancing is an example of a "Usage Policy". When a user configures load balancing they are defining a policy that specifies how to distribute client requests based on a set of criteria.

There are many ways to describe a policy or policies. For our purposes we will define a policy as composed of the following:

1. Policy Rules – 1 or more rules describing "what to do". A rule is made up of condition(s) and actions. Conditions can be as simple as "match anything" or as complex as "if source IP address 1.1.1.1 and it's 2:05". Likewise, actions can be as simple as "send to 2.2.2.2" or complex as "load balance using LRU between 3 NAS servers.)
2. Policy Domain – A collection of object(s) Policy Rules apply to. For example, suppose there was a policy that said "load balance using round robin". The collection of NAS servers being load balanced is the policy domain for the policy.

Policies can be nested to form complex policies.

2.6 Boot Sequence and Configuration

The MIC and other cards coordinate their actions during boot up configuration processing via System Service's Notify Service. These actions need to be coordinated in order to prevent the passing of traffic before configuration file processing has completed.

The other cards need to initialize with default values and set the state of their ports to "hold down" and wait for a "Config Complete" event from the MIC. Once this event is received the ports can be released and process traffic according to the current configuration. (Which may be default values if there were no configuration commands for the ports in the configuration file.)

FIGURE 14 illustrates this part of the boot up sequence and interactions between the MIC, S2 Notify and other cards.

There is an error condition in this sequence where the card never receives the "Config Complete" event. Assuming the software is working properly than this condition is caused by a hardware problem and the ports on the cards will be held in the "hold down" state. If CSM/CSA is working properly than the MIC Mgmt Software will show the ports down or CPCM might detect that the card is not responding and notify the MIC. In any case there are several ways to learn about and notify users about the failure.

3. LIC Software

The LIC (Lan Interface Card) consists of LAN Ethernet ports of 10/100/1000 Mbps variety. Behind the ports are 4 network engine processors. Each port on a LIC will behave like a layer 2 and layer 3 switch. The functionality of switching and intelligent forwarding is referred to herein as IFF – Intelligent Forwarding and Filtering. The main purpose of the network engine processors is to forward packets based on Layer 2, 3, 4 or 5 information. The ports will look and act like router ports to hosts on the LAN. Only RIP will be supported in the first release, with OSPF to follow.

3.1 VLANs

The box will support port based VLANs. The division of the ports will be based on configuration and initially all ports will belong to the same VLAN. Alternative practices of the invention can include VLAN classification and tagging, including possibly 802.1p and 802.1Q support.

3.1.1 Intelligent Filtering and Forwarding (IFF)

The IFF features are discussed in greater detail below. Layer 2 and layer 3 switching will take place inside the context of IFF. Forwarding table entries are populated by layer 2 and 3 address learning. If an entry is not known the packet is sent to the IP routing layer and it is routed at that level.

3.2 Load Balance Data Flow

NFS load balancing will be supported within a SANStream chassis. Load balancing based upon VIRUTAL IP addresses, content and flows are all possible.

The SANStream box will monitor the health of internal NFS servers that are configured as load balancing servers and will notify network management of detectable issues as well as notify a disk management layer so that recovery may take place. It will in these cases, stop sending requests to the troubled server, but continue to load balance across the remaining NFS servers in the virtual domain.

3.3 LIC – NAS Software

3.3.1 Virtual Storage Domains (VSD)

FIGURE 15 provides another VSD example. The switch system of the invention is designed to support, in one embodiment, multiple NFS and CIFS servers in a single device that are exported to the user as a single NFS server (only NFS is supported on the first release). These servers are masked under a single IP address, known as a Virtual Storage Domain (VSD). Each VSD will have one to many connections to the network via a Network Processor (NP) and may also have a pool of Servers (will be referred to as "Server" throughout this document) connected to the VSD via the fabric on the SRC card.

Within a virtual domain there are policy domains. These sub-layers define the actions needed to categorize the frame and send it to the next hop in the tree. These polices can define a large range of attributes in a frame and then impose an action (implicit or otherwise). Common polices may include actions based on protocol type (NFS, CIFS, etc.) or source and destination IP or MAC address. Actions may include implicit actions like forwarding the frame on to the next policy for further processing, or explicit actions such as drop.

FIGURE 15 diagrams a hypothetical virtual storage domain owned by Fred's ASP. In this example Fred has the configured address of 1.1.1.1 that is returned by the domain name service when queried for the domain's IP address. The next level of configuration is the policy domain.

When a packet arrives into the Pirus box from a router port it is classified as a member of Fred's virtual domain because of its destination IP address. Once the virtual domain has been determined its configuration is loaded in and a policy decision is made based on the configured policy. In the example above lets assume an NFS packet arrived. The packet will be associated with the NFS policy domain and a NAT (network address translation – described below) takes place, with the destination address that of the NFS policy domain. The packet now gets associated with the NFS policy domain for Yahoo. The process continues with the configuration of the NFS policy being loaded in and a decision being made based on the configured policy. In the example above the next decision to be made is whether or not the packet contains the gold, silver, or bronze service. Once that determination is made (let's assume the client was identified as a gold customer), a NAT is performed again to make the destination the IP address of the Gold policy domain. The packet now gets associated with the Gold policy domain. The process continues with the configuration for the Gold policy being loaded in and a decision being made based on the configured policy. At this point a load balancing decision is made to pick the best server to handle the request. Once the server is picked, NAT is again performed and the destination IP address of the server is set in the packet. Once the destination IP address of the packet becomes a device configured for load balancing, a switching operation is made and the packet is sent out of the box.

The implementation of the algorithm above lends itself to recursion and may or may not incur as many NAT steps as described. It is left to the implementer to short cut the number of NAT's while maintaining the overall integrity of the algorithm.

FIGURE 15 also presents the concept of port groups. Port groups are entities that have identical functionality and are members of the same virtual domain. Port group members provide a service. By definition, any member of a particular port group, when presented with a request, must be able to satisfy that request. Port groups may have routers, administrative entities, servers, caches, or other Pirus boxes off of them.

Virtual Storage Domains can reside across slots but not boxes. More than one Virtual Storage Domain can share a Router Interface.

3.3.2 Network Address Translation (NAT)

NAT translates from one IP Address to another IP Address. The reasons for doing NAT is for Load Balancing, to secure the identity of each Server from the Internet, to reduce the number of IP Addresses purchased, to reduce the number of Router ports needed, and the like.

Each Virtual Domain will have an IP Address that is advertised thru the network NP ports. The IP Address is the address of the Virtual Domain and NOT the NFS/CIFS Server IP Address. The IP Address is translated at the Pirus device in the Virtual Storage Domain to the Server's IP Address. Depending on the Server chosen, the IP Address is translated to the terminating Server IP Address.

For example, in FIGURE 15, IP Address 100.100.100.100 would translate to 1.1.1.1, 1.1.1.2 or 1.1.1.3 depending on the terminating Server.

3.3.3 Local Load Balance (LLB)

Local load balancing defines an operation of balancing between devices (i.e. servers) that are connected directly or indirectly off the ports of a Pirus box without another load balancer getting involved. A lower-complexity implementation would, for example, support only the balancing of storage access protocols that reside in the Pirus box.

Load Balancing Order of Operations:

In the process of load balancing configuration it may be possible to define multiple load balancing algorithms for the same set of servers. The need then arises to apply an order of operations to the load balancing methods. They are as follows in the order they are applied:

1) Server loading info, Percentage of loading on the servers Ethernet, Percentage of loading on the servers FC port, SLA support, Ratio Weight rating
2) Round Trip Time, Response time, Packet Rate, Completion Rate

3) Round Robin, Least Connections, Random

Load balancing methods in the same group are treated with the same weight in determining a servers loading. As the load balancing algorithms are applied, servers that have identical load characteristics (within a certain configured percentage) are moved to the next level in order to get a better determination of what server is best prepared to receive the request. The last load balancing methods that will be applied across the servers that have the identical load characteristics (again within a configured percentage) are round robin, least connection and random.

File System Server Load Balance (FSLB):

The system of the invention is intended to provide load balancing across at least two types of file system servers, NFS and CIFS. . NFS is stateless and CIFS is stateful so there are differences to each method. The goal of file system load balancing is not only to pick the best identical server to handle the request, but to make a single virtual storage domain transparently hidden behind multiple servers.

NFS Server Load Balancing (NLB):

NFS is mostly stateless and idempotent (every operation returns the same result if it is repeated). This is qualified because operations such as READ are idempotent but operations such as REMOVE are not. Since there is little NFS server state as well as little NFS client state transferred from one server to the other, it is easy for one server to assume the other server's functions. The protocol will allow for a client to switch NFS requests from one server to another transparently. This means that the load balancer can more easily maintain an NFS session if a server fails. For example if in the middle of a request a server dies, the client will retry, the load balancer will pick another server and the request gets fulfilled (with possibly a file handle NAT), after only a retry. If the server dies between requests, then there isn't even a retry, the load balancer just picks a new server and fulfills the request (with possibly a file handle NAT).

When using NFS managers it will be possible to set up the load balancer to load across multiple NFS servers that have identical data, or managers can set up load balancing to segment the balancing across servers that have unique data. The latter requires virtual domain configuration based on file requested (location in the file system tree) and file type. The former requires a virtual domain and minimal other configuration (i.e. load balancing policy).

The function of Load Balance Data Flow is to distribute the processing of requests over multiple servers. Load Balance Data Flow is the same as the Traditional Data Flow but the NP statistically determines the load of each server that is part of the specified NFS request and forwards the request based on that server load. The load-balancing algorithm could be as simple as round robin or a more sophisticated administrator configured policy.

Server load balance decisions are made based upon IP destination address. For any server IP address, a routing NP may have a table of configured alternate server IP addresses that can process an HTTP transaction. Thus multiple redundant NFS servers are supported using this feature.

TCP based server load balance decisions are made within the NP on a per connection basis. Once a server is selected through the balancing algorithm all transactions on a persistent TCP connection will be made to the same originally targeted server. An incoming IP message's source IP address and IP source Port number are the only connection lookup keys used by a NP.

For example, suppose a URL request arrives for 192.32.1.1. The Router NP processor's lookup determines that server 192.32.1.1 is part of a Server Group (192.32.1.1, 192.32.1.2, etc.). The NP decides which Server Group to forward the request to via user-configured algorithm. Round-Robin, estimated actual load, and current connection count are all candidates for selection algorithms. If TCP is the transport protocol, the TCP session is then terminated at the specified SRC processor.

UDP protocols do not have an opening SYN exchange that must be absorbed and spoofed by the load balancing IXP. Instead each UDP packet can be viewed as a candidate for balancing. This is both good and bad. The lack of opening SYN simplifies part of the balance operation, but the effort of balancing each packet could add considerable latency to UDP transactions.

In some cases it will be best to make an initial balance decision and keep a flow mapped for a user configurable time period. Once the period has expired an updated balance decision can be made in the background and a new balanced NFS server target selected.

In many cases it will be most efficient to re-balance a flow during a relatively idle period. Many disk transactions result in forward looking actions on the server (people who read the 1st half of a file often want the 2nd half soon afterwards) and rebalancing during active disk transactions could actually hurt performance.

An amendment to the "time period" based flow balancing described above would be to arm the timer for an inactivity period and re-arm it whenever NFS client requests are received. A longer inactivity timer period could be used to determine when a flow should be deleted entirely rather than re-balanced.

TCP and UDP – Methods of balancing:

NFS can run over both TCP and UDP (UDP being more prevalent). When processing UDP NFS requests the method used for psuedo-proxy of TCP sessions does not need to be employed. During a UDP session, the information to make a rational load balancing decision can be made with the first packet.

Several methods of load balancing are possible. The first and simplest to implement is load balancing based on source address – all requests are sent to the same server for a set period of time after a load balancing decision is made to pick the best server at the UDP request or the TCP SYN.

Another method is to load balance every request with no regard for the previous server the client was directed to. This will possibly require

obtaining a new file handle from the new server and NATing so as to hide the file handle change from the client. This method also carries with it more overhead in processing (every request is load balanced) and more implementation effort, but does give a more balanced approach.

Yet another method for balancing NFS requests is to cache a "next balance" target based on previous experience. This avoids the overhead of extensive balance decision making in real time, and has the benefit of more even client load distribution.

In order to reduce the processing of file handle differences between identical internal NFS servers, all disk modify operations will be strictly ordered. This will insure that the inode numbering is consistent across all identical disks.

Among the load balancing methods that can be used (others are possible) are:

- Round Robin
- Least Connections
- Random (lower IP-bits, hashing)
- Packet Rate (minimum throughput)
- Ratio Weight rating
- Server loading info and health as well as application health
- Round Trip Time (TCP echo)
- Response time

Write Replication:

NFS client read and status transactions can be freely balanced across a VLAN family of peer NFS servers. Any requests that result in disk content modification (file create, delete, set-attributes, data write, etc.) must be replicated to all NFS servers in a VLAN server peer group.

The Pirus Networks switch fabric interface (SFI) will be used to multicast NFS modifications to all NFS servers in a VLAN balancing peer group. All NFS client requests generate server replies and have a unique

transaction ID. This innate characteristic of NFS can be used to verify and confirm the success of multicast requests.

At least two mechanisms can be used for replicated transaction confirmation. They are "first answer" and quorum. Using the "first answer" algorithm an IXP would keep minimal state for an outstanding NFS request, and return the first response it receives back to the client. The quorum system would require the IXP to wait for some percentage of the NFS peer servers to respond with identical messages before returning one to the client.

Using either method, unresponsive NFS servers are removed from the VLAN peer balancing group. When a server is removed from the group the Pirus NFS mirroring service must be notified so that recovery procedures can be initiated.

A method for coordinating NFS write replication is set forth in FIGURE 16, including the following steps: check for NFS replication packet; if yes, multicast packet to entire VLAN NFS server peer group; wait for 1st NFS server reply with timeout; send 1st server reply to client; remove unresponsive servers from LB group and inform NFS mirroring service. If not an NFS replication packet, load balance and unicast to NFS server.

3.3.3 Load Balancer Failure Indication:

When a load balancer declares that a peer NFS server is being dropped from the group the NFS mirroring service is notified. A determination must be made as to whether the disk failure was soft or hard.

In the case of a soft failure a hot synchronization should be attempted to bring the failing NFS server back online. All NFS modify transactions must be recorded for playback to the failing NFS server when it returns to service.

When a hard failure has occurred an administrator must be notified and fresh disk will be brought online, formatted, and synchronized.

CIFS Server Load Balancing:

CIFS is stateful and as such there are fewer options available for load balancing. CIFS is a session-oriented protocol; a client is required to log on to a server using simple password authentication or a more secure cryptographic challenge. CIFS supports no recovery guarantees if the session is terminated through server or network outage. Therefore load balancing of CIFS requests must be done once at TCP SYN and persistence must be maintained throughout the session. If a disk fails and not the CIFS server, then a recovery mechanism can be employed to transfer state from one server to another and maintain the session. However if the server fails (hardware or software) and there is no way to transfer state from the failed server to the new server, then the TCP session must be brought down and the client must reestablish a new connection with a new server. This means relogging and recreating state in the new server.

Since CIFS is TCP based the balancing decision will be made at the TCP SYN. Since the TCP session will be terminated at the destination server, that server must be able to handle all requests that the client believes exists under that domain. Therefore all CIFS servers that are masked by a single virtual domain must have identical content on them. Secondly data that spans an NFS server file system must be represented as a separate virtual domain and accessed by the client as another CIFS server (i.e. another mount point).

Load balancing will support source address based persistence and send all requests to the same server based on a timeout since inactivity. Load balancing methods used will be:

- Round Robin
- Least Connections
- Random (lower IP-bits, hashing)
- Packet Rate (minimum throughput)
- Ratio Weight rating
- Server loading info and health as well as application